




Docker

Why?



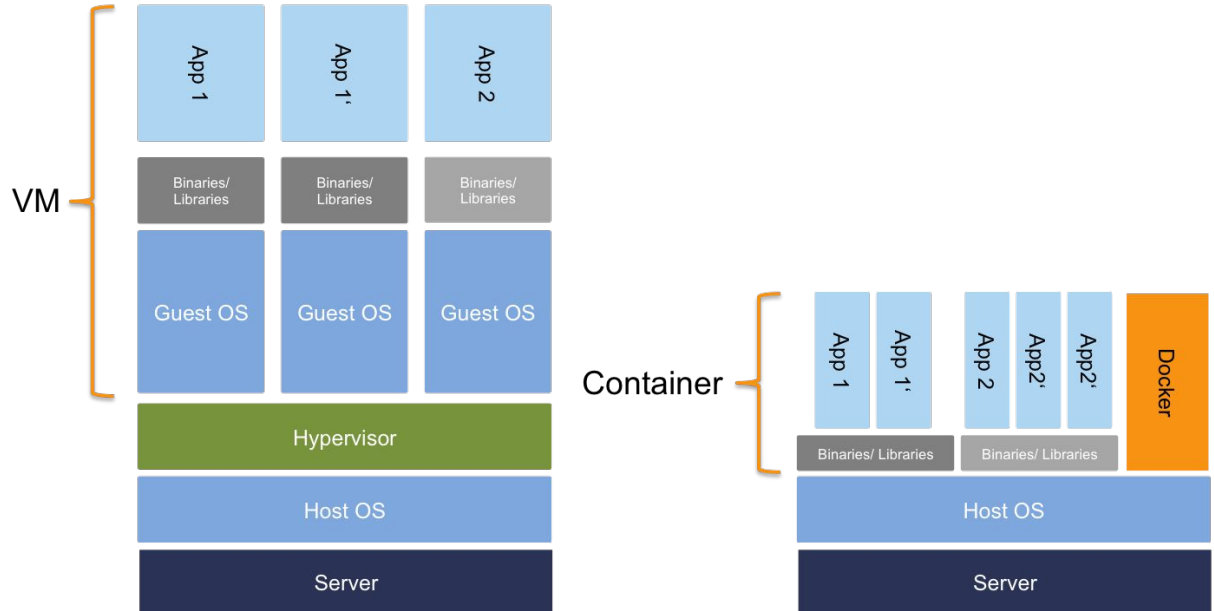
What is Docker?

- It is essentially an suite of tools for application isolation in Linux.
- Docker is a tool for automating the deployment of applications as portable containers.
- Containers allow you to automate the installation of dependencies and system configuration.

Images and Containers

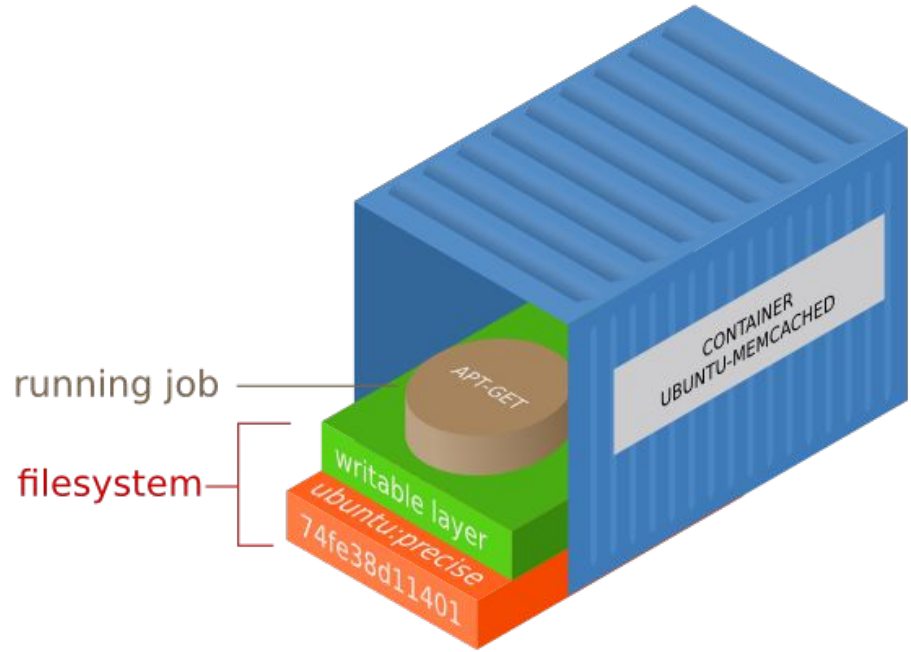
- A container is launched by running an image.
- An image is an executable package that includes everything needed to run the application including runtime, libraries, and configuration files.

Virtualisierung: Virtuelle Maschinen vs. Docker-Container



Containers

- A Docker container is an image with a read/writeable layer on top of several read-only layers.
- The middle layers are called intermediate image and are generated during Docker build.
- Typically, COPY and RUN commands translate into a single layer.



Basic Dockerfile

```
FROM python
```

```
WORKDIR /app
```

```
COPY . /app
```

```
RUN pip install --trusted-host pypi.python.org -r requirements.txt
```

```
EXPOSE 80
```

```
ENV NAME World
```

```
CMD ["python", "app.py"]
```

Viewing Layers

```
docker history <image>
```

```
$ docker history expressweb
```

| IMAGE | CREATED | CREATED BY | SIZE |
|--------------|------------|----------------------------------|----------|
| fdd93d9c2c60 | 2 days ago | /bin/sh -c CMD ["npm" "start"] | 0 B |
| e9539311a23e | 2 days ago | /bin/sh -c EXPOSE 8080/tcp | 0 B |
| 995a21532fce | 2 days ago | /bin/sh -c COPY dir:50ab47bff7 | 760 B |
| ecf7275feff3 | 2 days ago | /bin/sh -c npm install | 3.439 MB |
| 334d93a151ee | 2 days ago | /bin/sh -c COPY file:551095e67 | 265 B |
| 86c81d89b023 | 2 days ago | /bin/sh -c WORKDIR /usr/src/app | 0 B |
| 7184cc184ef8 | 2 days ago | /bin/sh -c mkdir -p /usr/src/app | 0 B |
| 530c750a346e | 2 days ago | /bin/sh -c CMD ["node"] | 0 B |

Building a Docker App

Docker build

Docker image ls

Docker run <imagename>

Good Practices

- Design your containers so they can be stopped, destroyed, and rebuilt with minimal down time and manual intervention.
- Combine RUN, COPY, and ADD operations as these create layers to avoid creating unnecessary layers.
- Always use the Alpine Image if possible.
 - It is optimized for Docker and minimizes layer depth
- Split long RUN statements into multiple lines to make the Dockerfile more readable
- Avoid apt-get upgrade
 - We can assume that the base image is reasonably up to date.
- Always do apt update

Run Statements

```
RUN set -ex \  
    ; command 1 \  
    ; command 2 \  
    ; final_command
```

Follow this style

Organize RUN statements to optimize build time

- Order them from less frequently changed to more frequently changed
 - This ensures the build cache is reusable
- Suggested Order:
 - Install build tools
 - Install or update library dependencies
 - Generate or install your applications
 - Set configuration files

Cache Invalidation

```
$ docker history expressweb
```

| IMAGE | CREATED | CREATED BY | SIZE |
|--------------|------------|----------------------------------|----------|
| fdd93d9c2c60 | 2 days ago | /bin/sh -c CMD ["npm" "start"] | 0 B |
| e9539311a23e | 2 days ago | /bin/sh -c EXPOSE 8080/tcp | 0 B |
| 995a21532fce | 2 days ago | /bin/sh -c COPY dir:50ab47bff7 | 760 B |
| ecf7275feff3 | 2 days ago | /bin/sh -c npm install | 3.439 MB |
| 334d93a151ee | 2 days ago | /bin/sh -c COPY file:551095e67 | 265 B |
| 86c81d89b023 | 2 days ago | /bin/sh -c WORKDIR /usr/src/app | 0 B |
| 7184cc184ef8 | 2 days ago | /bin/sh -c mkdir -p /usr/src/app | 0 B |
| 530c750a346e | 2 days ago | /bin/sh -c CMD ["node"] | 0 B |

ADD vs COPY

- Copy can only copy files into the container.
- Prefer copy whenever possible.
- Add has many functions and can sometimes lead to unexplained behavior.
- Add is most commonly used for copying a tarball into a container and unzipping it.

Extracting a tarball:

Add Method:

```
ADD resources/jdk-7u79-linux-x64.tar.gz /usr/local/
```

Copy/Run Method

```
COPY resources/jdk-7u79-linux-x64.tar.gz /tmp/  
RUN tar -zxvf /tmp/jdk-7u79-linux-x64.tar.gz -C /usr/local  
RUN rm /tmp/jdk-7u79-linux-x64.tar.gz
```

Work Directory

The WORKDIR instruction sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it in the Dockerfile.

- For clarity, always use absolute paths.
- Avoid using RUN cd ... ; do something

Shell vs Exec Form

- Shell form: <instruction> command
 - RUN apt-get install python3
CMD echo "Hello world"
ENTRYPOINT echo "Hello world"
- Exec form <instruction> ["executable", "param1", "param2", ...]
 - ENV name John Dow
ENTRYPOINT ["/bin/echo", "Hello, \$name"]
Output: Hello, \$name
 - No shell processing
 - preferred form for cmd and entrypoint

CMD

- CMD instruction should be used to run the software contained by your image along with any parameters.
- Specifically, use CMD to set the default command and parameters which will be run by your container.
- Your entry point will still be called.

Three forms of CMD

- CMD ["executable","param1","param2"]
 - preferred form
- CMD ["param1","param2"]
 - sets additional parameters to entry point
- CMD command param1 param2
 - shell form
 - Please avoid

Entrypoint

- The entry point executes the commands to run on the start of the container.
- Dockers best practices guidelines acknowledge two styles of Entrypoints for well designed containers.
 - Example in two slides
- Has two forms as well: shell form and exec form
- Do not use entrypoint to build your container

Entrypoint Forms

- `ENTRYPOINT ["executable", "param1", "param2"]` (*exec* form, preferred)
- `ENTRYPOINT command param1 param2` (*shell* form)

'Main Command' Entrypoint

- Use the entrypoint to set the images 'main command,' effectively making the image act as if it is the binary.
- Okay for very simple, single purpose, containers.
- Effectively bad design because: you cannot interactively enter the container for debugging easily.

Dockerfile:

```
ENTRYPOINT ["openssl"]
```

```
CMD ["--help"]
```

Entrypoint to Configure Container

```
ENTRYPOINT [ "docker-entrypoint.sh" ]  
CMD [ "nginx", "-g", "daemon off;" ]  
# docker-entrypoint.sh  
#!/bin/sh  
set -ex  
if [ "$1" = "nginx" ]; then  
    update-nginx-conf.sh  
fi  
exec "${@}"
```

Entrypoint Practices Explained

- This way you can use the container interactively without redefining the entry point.
- The entry point only configures if you are intending to run the service, otherwise allow interactive entry.
- We are using CMD to provide arguments to ENTRYPOINT.
- Use ENTRYPOINT to execute the arguments passed to it by CMD.
- Entrypoint should be used only for configuring a container not building the container.
- Dockerfiles should specify at least one CMD or ENTRYPOINT commands
- The last line of a entrypoint should always be exec.

How to write good entry points

- How to write good entry points
- Use good scripting practices
- Keep them simple.
- Don't try and build the image in your entrypoint. Only configure it.
 - Building stuff should go into the docker file in an RUN clause
- The last line in your entry point should always be to execute your application or another binary.
 - This makes the process be PID 1 in the container
 - You want your application to be pid 1 because this is where docker were docker will send all of the unix signals
 - usually exec all params: `exec "${@}"`



Conclusion

